WHAT IS CLAIMED IS:

1. In a computer system adapted to executing binary translated code, a method for responding to an exception comprising the steps of:

defining a plurality of recovery points in the binary translated code, each of said plurality of recovery points comprising an executable instruction;

prior to execution of said executable instruction, saving the address of said executable instruction;

executing subsequent instructions;

detecting an exception;

invoking an exception handler;

reconfiguring the state of the computer system to reflect the state of the computer system at said executable instruction;

using run-time information, translating foreign code beginning at an instruction corresponding to said executable instruction; and

determining if said exception re-occurs.

2. The method of claim 1 further comprising the step of, upon resolving the exception, returning to said binary translated code following the recovery point that correspond to said executable instruction.

3. The method of claim 2 further comprising the step of terminating execution of said binary translated code if said exception handler is unable to resolve the exception.

4. The method of claim 1 further comprising the step of defining said plurality of recovery points during binary translation of foreign code.

5. The method of claim 5 wherein said step of defining said plurality of recovery points is performed at run-time.

6. The method of claim 4 wherein said instructions comprise a plurality of operations and said defining step further comprises the step of optimizing the sequence of execution of said operations.

7. The method of claim 6 wherein said optimizing step further comprises the steps of:

maintaining documentation describing intermediate results of said operations; and

committing the values of said plurality of registers to documentation prior to execution of said executable instruction.

8. The method of claim 7 wherein said invoking step further comprises the steps of:

recovering the address of said executable instruction;

determining the location of said set of registers; and

invoking a binary translator to re-translate a sequence of foreign code corresponding to said executable instruction and said subsequent instructions, said binary translator adapted to generate an in-order sequence of binary translated code corresponding to said foreign code.

9. The method of claim 8 wherein said invoking step further comprises the steps of: committing the state of said computer system to said documentation at a next to occur recovery point.

10. In a computer system adapted for executing optimized binary translated code while maintaining precise exception order, said computer system comprising:

a processing unit for executing a sequence of instructions, each of said instructions comprising a plurality of operations that may be executed in parallel; each of said operations having at least one operand; said processing unit adapted to detect the occurrence of an exception;

a register set for holding results of said operations and the state of said computer system as determined by execution of said operations and for storing the value of said at least one operand prior to executing said each of said instructions; and

an exception handler adapted to access said register set to recover the value of said operand and to generate an intermediate state representative of the state of said processing unit prior to the detection of the exception.

11. The system of claim 10 further comprising means for transferring the contents of said register set to a storage location prior to execution of a subsequent instruction.

12. The system of claim 10 wherein said exception handler further comprises a set of documentation identifying a memory location for said contents.

13. The system of claim 12 wherein said exception handler further comprises means for recovering from said exception by translating a portion of foreign code corresponding to said each of said instructions.

14. In a computer system adapted for executing optimized binary translated code while maintaining precise exception order of foreign code, a method for recovering from a detected exception comprising the steps of:

documenting information representative of the state of said computer system;

executing at least a first instruction, said first instruction having a plurality of operations executed in speculative mode;

executing a second instruction;

detecting whether the execution of said second instruction generated an exception;

upon detection of an exception, transferring information to a register set;

repeating execution of said at least a first instruction; and

transferring program execution to an exception handler.

15. The method of claim 14 wherein said register set comprises a set of general purpose registers.

16. The method of claim 14 wherein said documenting information is stored in a portion of said register set.

17. In a computer system adapted for executing optimized binary translated code while maintaining precise exception processing of foreign code comprising the steps of:

A) Executing a first sequence of instructions in a speculative mode where each of said instructions comprise a plurality of operations which may be executed in parallel;

B) Saving the address of at least one of the instructions in said sequence of instructions in a storage location;

C) Saving the status of said computer system in a second storage location each time said at least one of the instructions is executed;

D) Executing a next to occur instruction in said first sequence of instructions;

E) Detecting whether the execution of said next to occur instruction generated an exception;

F) If an exception is detected, recovering said address from said storage location;

G) Recovering said system status from said second storage location;

H) Repeating the execution of said first sequence of instructions beginning at said at least one of the instructions whereby each of said operations are executed in-order; and

I) If said first sequence of instructions are executed and an exception is not detected:

J) Committing said status of said computer system to a register set; and

K) Repeating steps A) through F) for a second sequence of instructions.

18. The method of claim 17 wherein said registers set comprise a set of registers configured to mirror a foreign register set.

19. The method of claim 17 wherein speculative mode comprises the steps of Determining if an operation could generate an exception;

Generating a diagnostic operand when an exception while in speculative mode;

Denote that said diagnostic operand is a speculative value; and

If said diagnostic operand is not modified when said diagnostic operand is changed to a non-speculative value, invoking an exception.

20. A method of optimizing binary translated code by extracting the parallelism inherent to foreign code during the binary translation process; said optimizing comprising the sequential reordering of operations and memory access without violating precise exception order in binary translated code executing on a host platform having explicit parallelism architecture, said method comprising the steps of:

a) allocating a set of registers and a dedicated memory region for storing information regarding the location of a foreign register set in said register set;

b) designating a set of Recovery Points in said binary translated code where each Recovery Point in the binary translated code has correspondence with an instruction in the foreign code, and each Recovery Point is associated with a documentation set, saved on hard disk or in memory, that contains information where all foreign registers are located in the host registers in the optimized binary translated code;

c) responding to an exception between adjacent Recovery Points by re-invoked the interpretation of foreign instructions in sequential fashion by starting execution from the previous Recovery Point, i.e. foreign context (foreign registers and memory) must not be changed in the optimized binary translated code irretrievably.

21. The method of claim 20 further comprising the step of designating each memory write operation into the foreign memory space as a Recovery Point.

22. The method of claim 20 further comprising the step of designating at least one selected operation corresponding to an instruction in foreign memory space as a Recovery Point where said selected operation changes the state of the foreign memory space in an unrecoverable manner.

23. The method of claim 20 further comprising the step of maintaining the state of said computer system corresponding to the preceding Recovery Point until a next to occur Recovery Point is reached.

24. The method of claim 23 wherein said maintaining step comprises the step of saving information sufficient to restore the contents of the foreign registers at the preceding Recovery Point.

25. The method of claim 24 wherein said saving step comprises the step of saving selected register names comprising foreign registers.

26. The method of claim 25 further comprising the step of renaming said selected registers for optimization of said binary translated code after passing said next to occur Recovery Point.

27. The method of claim 23 further comprising the step of marking each Recovery Point in the binary translated code with a "SetIP" operation.

28. The method of claim 23 further comprising the step of saving a host address of the wide instruction for the marked Recovery Point in a Recovery Point Register.

29. In a computer system adapted to executing a computer program comprising binary translated code, a method for reconstructing the cause of an exception comprising the steps of:

temporarily preserving register data and system status information before executing instructions that will calculate a variable;

invoking an exception handler prior to generating side effects;

responding to said exception with said exception handler using said temporarily preserved register data to redirect operation of said computer program

30. The method of claim 29 wherein operands in the temporary registers are retained until reassigned by another operation.

31. The method of claim 30 further comprising, in response to an exception, the step of recalculating operands by re-executing the instruction to recover side effects.

32. The method of claim 31 further comprising the step of allocating a portion of general-purpose registers in a register set to function as temporary registers.

33. The method of claim 32 further comprising the step of allocating a memory region for storing information regarding the location of a foreign register set in said register set.

34. A computer system adapted for executing a computer program comprising binary translated code comprising:

a first memory for storing foreign code;

a second memory for storing optimized binary translated code;

a recovery point register for storing the address of a selected instruction in said binary translated code;

a processor for executing binary translated code and adapted for detecting the occurrence of an exception; said processor coupled to said recovery point register and to said first and second memory;

an exception handler, coupled to said processor, for recovering the address stored in said recovery point register in response to detection of an exception by said processor, said address corresponding to one instruction in said foreign code; and

a set of documentation stored in said second memory for preserving register data and system status information of the state of said computer system prior to detection of said exception;

means for translating said foreign code to obtain sequential binary code corresponding to a portion of foreign code beginning at said one instruction in said foreign code.